Detecting Malware of Windows OS Using AI Classification for Image of Extracted Behavior Features

Kang Dongshik, Noor Aldeen Alhamedi
University of the Ryukyus, Okinawa, Japan

***Abstract*—Malware detection is crucial for protecting digital environments. Traditional methods involve static and dynamic analysis, but recent advancements leverage artificial intelligence (AI) to enhance detection accuracy. This study aims to improve malware detection by integrating dynamic malware analysis with AI-driven techniques. The primary challenge addressed is accurately classifying and detecting malware based on behavior extracted from isolated virtual machines. By analyzing 50 malware samples and 11 benign programs, we extract ten behavioral features such as process ID, CPU usage, and network connections. We employ text-based classification using feedforward neural networks (FNN) and recurrent neural networks (RNN), achieving accuracy rates of 56% and 68%, respectively. Additionally, we convert the extracted features into grayscale images for image-based classification with a convolutional neural network (CNN), resulting in a higher accuracy of 70.1%. This multi-modal approach, combining behavioral analysis with AI, not only enhances detection accuracy but also provides a comprehensive understanding of malware behavior compared to competing methods.**

***Keywords*—Malware analysis; dynamic-based analysis; image classification; malware behavior extraction; text**

I. Introduction

Recently, the number, severity, sophistication of malware attacks, and cost of malware inflicts on the world economy have been increasing exponentially. Attacks with these kinds of software have disastrous effects and cause considerable material damage to individuals, private companies, and governments' assets. Thus, malware should be detected before damaging the important assets in the company [1]. The primary motivation for this research stems from the need to enhance existing detection mechanisms to keep pace with the constantly changing threat landscape. With traditional analysis methods, we aim to significantly improve the detection and classification accuracy of malicious software. One of the key advantages of our approach is the combination of dynamic-based malware analysis with AI-driven techniques. This allows for a more comprehensive understanding of malware behavior. This hybrid approach not only improves detection rates but also enhances the ability to accurately classify and understand the nature of malware.

There are two main techniques for analyzing malware static and dynamic-based analysis. Static-based analysis examines the malware code without actually executing it. This by integrating advanced artificial intelligence (AI) techniques can provide information about suspicious functions, network activity, impacted files, etc. Dynamic-based analysis executes the malware code in an isolated environment to observe its runtime behavior. This provides insight into the full impact of the malware. A key benefit of static-based analysis is the ability to thoroughly inspect malware code using techniques like disassembly and decompilation to identify suspicious functions related to replication, propagation, payload activation, and more [2]. The static techniques help reveal overall structure, dependencies, triggers for malicious events, and obfuscation attempts. However, lacking runtime behavior, static-based analysis cannot confirm the real impact of suspected capabilities. Complex packing or encryption techniques also limit code inspection. Other hand, the dynamic-based analysis provides direct observation of malware behavior in action by executing it and monitoring the resulting activity.

Dynamic-based analysis confirms suspected functions based on static clues and captures full infection chains showing the progression and end objectives of malware according to case studies by [3]. Dynamic monitoring of memory access, network calls, system API usage, and more creates a comprehensive picture. Additionally, dynamic-based analysis is particularly effective in identifying and analyzing newly emerging malware strains. As it focuses on the runtime behavior, it is better equipped to handle polymorphic and metamorphic malware that may change its form to evade static-based analysis techniques. Leveraging AI models for the analysis of malware code or the study of malware behavior has significantly contributed to the detection of malware in recent years. Numerous AI models have been integrated into static or dynamic approaches to augment both the malware detection rate and feature extraction processes. Despite the notable progress in the field of AI, these models still face various challenges. This research will use many models of AI to detect malware.

Robust malware analysis faces numerous obstacles. The sheer volume of malware proliferating at a rapid pace presents a formidable challenge in comprehensively examining this ever-expanding threat landscape. Additionally, malware authors employ sophisticated obfuscation tactics, such as code interchange, amalgamation, register reassignment, null insertion, and subroutine reordering [3], purposefully designed to evade detection by anti-malware systems. Despite decades of development, these security solutions still exhibit high false positive rates, undermining their accuracy.

Moreover, certain malware strains possess the ability to identify virtualized environments, resulting in altered or ceased execution, hindering effective analysis. The evasion techniques employed by malware necessitate lengthy detection times, potentially ranging from minutes to hours depending on the specific malware variant, during which systems remain vulnerable to compromise. Furthermore, the ambiguity surrounding API calls, as both malicious and benign software may legitimately invoke common APIs, complicates the process of distinguishing malware based on API usage patterns.

These factors, including the immense scale, obfuscation methods, virtual environment detection capabilities, delayed identification timelines, and the dual usage of APIs, collectively contribute to the arduous nature of robust malware analysis, necessitating the development of advanced techniques to overcome these challenges effectively. The juxtaposition of text classification and image classification in the analysis of extracted behavior. It underscores that a nuanced understanding of program nature, distinguishing between benign and malicious entities, can be achieved through thorough behavior analysis. The model primarily relies on the extraction of malware features. Within the developed script, two distinct observers play a crucial role. The first observer extracts the entirety of the process, encompassing its characteristics, as well as details related to internet connections. The second observer is tasked with monitoring any file creation specifically linked to the malware. The experimental framework involves the extraction of 10 distinct features through the monitoring of behaviors within an isolated Virtual Machine. Python libraries such as psutil, subprocess, wmi, watchdog, time, json, and os were employed to develop functions responsible for observing malware behavior and subsequently extracting pertinent information to a JSON file. The extracted features encompassed critical aspects such as process ID, process name, username, CPU percentage.

II. Related Work

Artificial Intelligence (AI) has emerged as a powerful tool in this ongoing struggle to detect and classify malware offering advanced capabilities in identifying and mitigating malware threats.

In a study [4], the third paper analyzes different classical machine learning algorithms for malware detection - Random Forest, Support Vector Machine (SVM), grid search optimized SVM, and K-Nearest Neighbors (KNN). The goal is to validate the effectiveness of these models for detecting zero-day malware attacks. The dataset from Kaggle contained 19,611 PE files, with 14,599 malicious samples and 5,012 benign files with 77 numeric features. Three training/test splits were used. Various accuracy metrics were calculated: accuracy, F1-score, confusion matrix, precision, recall and Type I/II errors. Random Forest performed the best with 96% accuracy and 93% F1score, with low errors and fastest training time. Optimized SVM improved results significantly but slowed down execution. KNN also performed decently with simpler implementation. Analysis showed Random Forest has good prospects for realtime zero-day malware detection. The model can process 25,000 files per second. For deployment, more diverse input data covering different malware families is needed.

In study [5], the authors used convolutional neural networks (CNNs) for malware classification by visualizing malware programs as grayscale images. The images are generated from the bytecode of malware programs and classified using CNN architectures. They evaluate several well-known CNN models like AlexNet, ResNet, and VGG16 using transfer learning on a malware image dataset. They also propose a custom shallow CNN architecture that achieves 96% accuracy, but is faster to train than the other complex models. The customized CNN and transfer learning models are also tested as feature extractors, with the features fed into SVM and KNN classifiers. This achieves even better performance up to 99.4% accuracy. They set a new benchmark on the public BIG 2015 malware dataset. The proposed system combining CNN feature extraction + SVM classifier obtains state-of-the-art 99.4% accuracy in distinguishing between nine malware classes. Visualization and CNN-based classification is shown to be effective for malware detection. The approach is computationally efficient compared to static/dynamic-based analysis. Fusing different CNN model predictions can further improve performance.

In study [6], the authors used Support Vector Machines (SVMs) for malware analysis and classification. SVMs are supervised learning models that can analyze high-dimensional, sparse data and recognize patterns. The authors collect a heterogeneous malware dataset from a real threat database. The data has features like time, format, domain, and IP address. They visualize the dataset using techniques like scatter plots and radius visualization to understand correlations and structure before classification. An SVM model with a polynomial kernel is trained on the dataset to classify malware vs normal software. The model is validated using cross-validation, leave-one-out and random sampling. The SVM classifier achieves 93-95% accuracy, 97-98% sensitivity and 86-90% specificity on the malware dataset. Validation shows the model generalizes very well. The high-performance highlights that SVMs can effectively classify heterogeneous malware data gathered from computer networks and security systems.

In study [7], the paper proposes a deep learning framework for malware visualization and classification using convolutional neural networks (CNNs). The key aspects are: Malware files are converted into three image types - grayscale, RGB color, and Markov images. Markov images help retain global statistics of malware bytes. A Gabor filter approach is used to extract textures and discriminative features from the malware images. Two CNN models are used for classification – a custom 13-layer CNN and a pretrained 71-layer Xception CNN fine-tuned for malware images. The framework is evaluated on two public Windows malware image data sets, a custom Windows malware dataset, and a custom IoT malware dataset. Markov images provide the best results, with the fine-tuned Xception CNN achieving over 99% accuracy on multiple datasets. The computational efficiency is also better compared to prior works. The approach demonstrates effectiveness for real-time malware recognition and classification. The visualization and deep learning framework extracts features automatically without extensive feature engineering.

The framework's resilience against adversarial attacks is also analyzed by adding noise to test images. Some drop in accuracy is noticed, indicating scope for improvement. The current landscape underscores the significance of AI models as powerful tools for the analysis, classification, and detection of malware. These models can seamlessly integrate with both static and dynamic-based analysis, yielding noteworthy results that underscore their pivotal role in shaping the future of this field.

Arabo et al. [8] analyzed CPU and RAM usage patterns as potential indicators for detecting ransomware processes. Their findings suggested that while not the primary factors, monitoring CPU and RAM could complement other behavioral characteristics in identifying malicious processes. Regarding CPU usage, they observed variations that showed potential for distinguishing ransomware activities. Specifically, for the ViraLock ransomware sample, the maximum CPU usage peaked at 25% [1]. Such CPU spikes could potentially signify the initiation of encryption or other malicious operations by the ransomware. As for RAM consumption, the study found that ransomware samples generally exhibited low and relatively stable memory usage patterns. In the case of ViraLock, the maximum RAM usage was only around 2% [1]. However, the authors noted that while low RAM usage alone may not be a definitive indicator, it could be considered in combination with other behavioral factors. The researchers highlighted that while CPU and RAM usage showed some differences between ransomware and benign processes, the most significant distinguishing factor was abnormally high disk read/write activity [1]. Nonetheless, incorporating CPU and RAM monitoring alongside disk usage analysis could potentially enhance the accuracy and robustness of ransomware detection systems based on process behavior analysis.

III. Methodology

The current investigation is centered on the behavioral analysis within an isolated Windows environment in virtual machine for the purpose of detecting malware. To achieve this, a combination of Recurrent Neural Network (RNN) for text classification and Convolutional Neural Network (CNN) for image classification is employed to analyze the extracted data. Diverging from the methodologies outlined in previous studies [3], [6], and [7], the classification approach adopted here focuses on the inherent characteristics of the malware file itself. This is achieved through a comprehensive analysis of the malware binary file and, notably, by representing the malware file as an image utilizing various visualization techniques. In this research, the emphasis is on visualizing the malware's behavior and, subsequently, conducting analyses based on these visual representations and also analysis the extracted features as a text. The presented model offers a juxtaposition of text classification and image classification in the analysis of extracted behavior. It underscores that a nuanced understanding of program nature, distinguishing between benign and malicious entities, can be achieved through thorough behavior analysis.

The model primarily relies on the extraction of malware features. Within the developed script, two distinct observers play a crucial role. The first observer extracts the entirety of the process, encompassing its characteristics, as well as details related to internet connections. The second observer is tasked with monitoring any file creation specifically linked to the malware.

The experimental framework involves the extraction of 10 distinct features through the monitoring of behaviors within an isolated Virtual Machine. Python libraries such as psutil, subprocess, wmi, watchdog, time, json, and os were employed to develop functions responsible for observing malware behavior and subsequently extracting pertinent information to a JSON file. The extracted features encompassed critical aspects such as process ID, process name, username, CPU percentage.

The modules for this research were developed using TensorFlow and Keras, leveraging the Sequential model architecture. These tools enabled efficient construction and training of neural networks for malware detection, facilitating both text-based and image-based classification with enhanced accuracy through deep learning techniques. Fig. 1 shows proposed processing model.

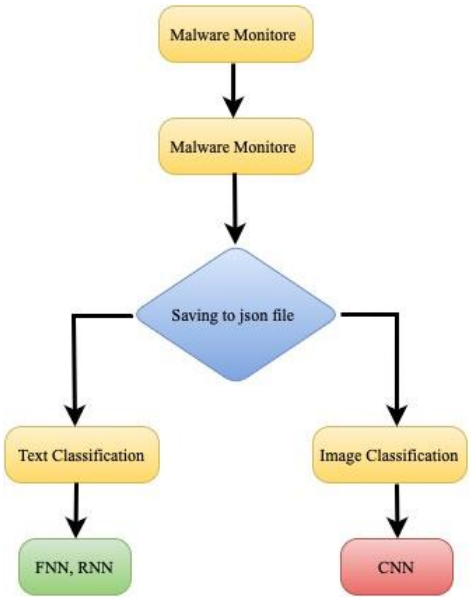

Fig. 1. Proposed processing.

Following the extraction of these features, the gathered information is stored in a JSON file (see Fig. 2) for further next step.

A. Text Analysis

The analytical process for the extracted features unfolded across two phases. Initially, the data underwent textual analysis, leveraging a simple feedforward neural network (FNN) model designed for binary classification using the Keras library to create a fully connected dense layer with 128 nodes. The output layer has 1 node and uses 'sigmoid' activation for binary classification. Subsequently, a recurrent neural network (RNN) model was employed to classify the same textual data, creates an embedding layer that transforms integer word indices to dense word vector representations.

```
0:
  label:          0
  pid:            48872
  name:           "4f97a7f893939680bf36ccc03af19cc2d9ae3e4c7696fefc79ff5750ace15bae.exe"
  username:       "WINDOWS-10\\vboxuser"
  cpu_usage:      "none"
  connections:    "[pconn(fd=-1, family=<AddressFamily.AF_INET: 2>, type=<SocketKind.SOCK_STREAM: 1>,
                  laddr=addr(ip='10.0.2.15', port=50603), raddr=addr(ip='34.117.59.81', port=443),
                  status='ESTABLISHED'), pconn(fd=-1, family=<AddressFamily.AF_INET: 2>,
                  type=<SocketKind.SOCK_STREAM: 1>, laddr=addr(ip='10.0.2.15', port=50602),
                  raddr=addr(ip='194.169.175.113', port=50500), status='ESTABLISHED')]"
  parent:         "none"
  child:          "[{'ExecutablePath
                  \\r\\r'}, {'C:\\\\Users\\\\vboxuser\\\\Desktop\\\\mal-
                  DB\\\\4f97a7f893939680bf36ccc03af19cc2d9ae3e4c7696fefc79ff5750ace15bae.exe'}]"
  execution:      "none"
  filecreated:
    0:            '{"file_path": "C:\\\\Users\\\\vboxuser\\\\AppData\\\\Local\\\\Microsoft\\\\Edge\\\\User
                  Data\\\\Cookies"}\n{"file_path": "C:\\\\Users\\\\vboxuser\\\\PycharmProjects
                  \\\\pythonProject\\\\venv\\\\Scripts\\\\mal-file_created00"}\n'
```


Fig. 2. Sample of Json file content connection details, parent process, child process, execution path, and created files.

B. Image Analysis

By transforming data into images, researchers can leverage the vast body of knowledge and advancements in image processing techniques, readily applicable to the analysis of the transformed data. This data-to-image transformation unlocks the power of CNNs for a wider range of analysis tasks, promoting deeper insights into complex datasets. So this research implements the power of CNN alongside with the behavior analysis Subsequent to the behavioral analysis, the extracted features underwent further evaluation through an image classification paradigm. A dedicated function was developed to transform these feature data into grayscale images. This transformative process involved the removal of associated labels, conversion of the data into binary numerical representations, subsequent transformation of these binary values into hexadecimal equivalents, and, finally, depiction of these hexadecimal values onto a 30*30 grayscale canvas.

The 30x30 size was empirically determined to balance information preservation and computational efficiency. Representing features as images enabled the utilization of convolutional neural networks (CNNs), which excel at capturing spatial patterns the extracted features underwent further evaluation through an image classification paradigm. This visual representation approach offered several key advantages. Firstly, it enabled leveraging powerful deep learning techniques like convolutional neural networks, adept at capturing spatial patterns invaluable for malware characterization. Secondly, transforming features into images facilitated uncovering intrinsic relationships and patterns obfuscated in the original data's raw representation. Thirdly, the image domain allowed seamless integration of transfer learning and pre-trained models, expediting the analysis process. Lastly, the visually interpretable nature of images could provide insights into the discriminative characteristics learned by the models, aiding explain ability. By combining dynamic monitoring with visual analytics, this multi-pronged approach offered a potent framework for comprehensive malware analysis and classification.

The dataset employed for experimentation comprised 50 instances of .EXE malware sourced from diverse families, obtained from the Malware Bazaar database, a freely accessible online repository. Additionally, 11 benign programs were included for comparative analysis. The monitoring process lasted three seconds for every malware instance, during which the monitoring code ran in the background, observing the processes and file creation activities of the malware. After the monitoring period, the code produced a JSON file containing the captured information. The dataset has been divided into 40 malware behavior and six benign program behavior for the training and 10 malware behavior and five benign program behavior for testing. Fig. 3 shows converting text to image process.

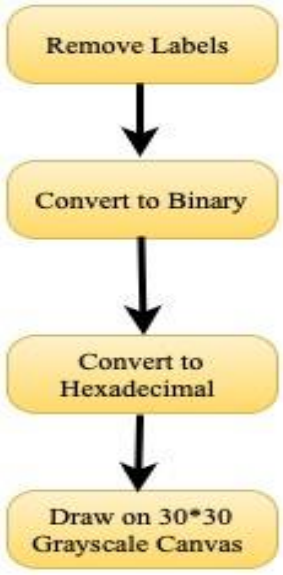

Fig. 3. Converting text to image.

IV. EXPERIMENTS

A. Text Analysis

The described FNN model exhibited an accuracy rate of 56% with a corresponding loss rate of 0.78. For the RNN model: It takes the vocabulary size equal to 32 and output dimensionality as arguments. Also LSTM layer models the sequential nature and long-range context of text. The output dense layers act as classifiers on top of LSTM representations. The model is compiled with binary cross entropy loss, adam optimizer and accuracy metric.

With epoch 100, yielding an improved accuracy rate of 68% with a reduced loss rate of 0.67.

B. Image Analysis

Convolutional Neural Networks (CNNs) have revolutionized image analysis due to their ability to extract intricate spatial features. However, their power can be extended to non-image data by transforming it into a suitable image representation. This approach offers several advantages: CNNs excel at automatically learning relevant features from images, circumventing the need for manual feature engineering, a time-consuming and potentially error-prone step in traditional analysis. Data transformation allows for the visualization of complex relationships between data points within the image domain. This empowers CNNs to identify subtle patterns that might be obscured in the raw data format. The experiment was done using two suggested models. The first model (Fig. 5) is simple and the second model is more complex both models are based on CNN. The simple model consists of:

- Conv2D layer: Performs 2D convolution with 32 filters and 3x3 kernel. Extracts spatial features from input image.
- MaxPool2D: Max pooling layer reduces dimensions to summarize the features detected by the convolution layer.
- Flatten: Flattens the pooled feature map into a 1D vector to prepare for fully-connected layers.
- Dense layers: Fully-connected layers that act as classifier on top of the extracted features. 64 nodes in first dense layer.

Output layer contains single node with 'sigmoid' activation for binary classification. This model takes input images of shape (30, 30, 1) indicating 30x30 grayscale images. Using this simple model over these grayscale pictures gives accuracy rate 70.1% with loss 0.67.

The second model also based on CNN with more complex architecture: The model then uses several convolutional layers (Conv2D) to extract features from the image. These layers apply filters (also called kernels) that slide across the image, detecting patterns and edges.

The first Conv2D layer has 256 filters, each of size 3x3. As the filter slides across the image, it performs element-wise multiplication between the filter weights and the corresponding pixel values in the image. The results are then summed and passed through an activation function (relu in this case) to introduce non-linearity. This process helps identify low-level features like edges, corners, and simple shapes. The subsequent Conv2D layers follow the same principle but with a different number of filters (128 and 64 in this example). These layers extract progressively more complex features based on the lower-level features detected earlier.

MaxPooling2D layers are inserted after some convolutional layers. These layers downsample the feature maps by taking the maximum value within a specific window (2x2 in this example). This helps reduce the number of parameters and computational cost while potentially capturing the most important features. Fig. 4 shows sample representation of the resultant images.

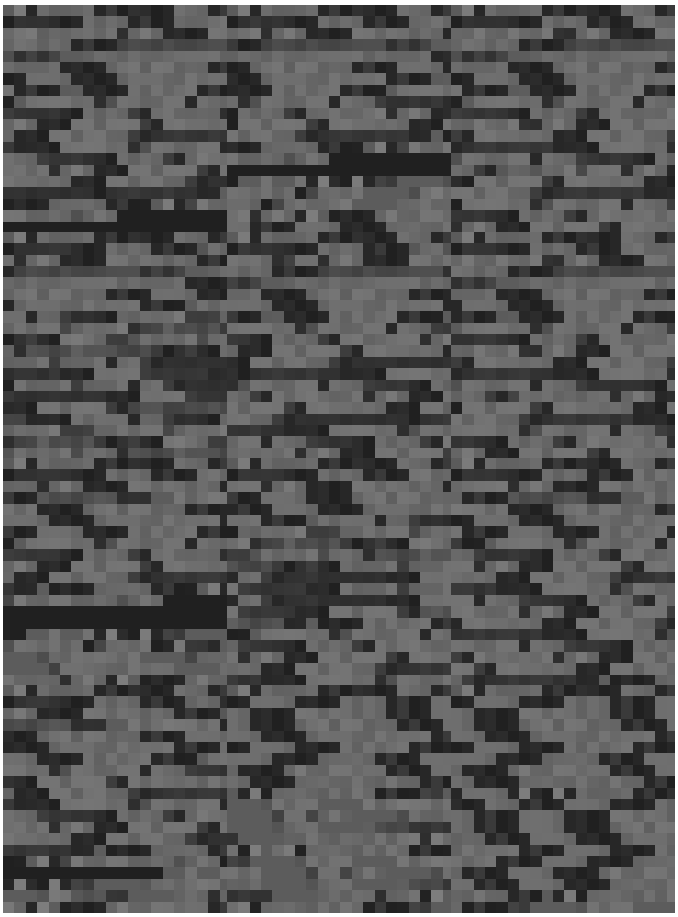

Fig. 4. A sample representation of the resultant images, offering a glimpse into their visual characteristics.

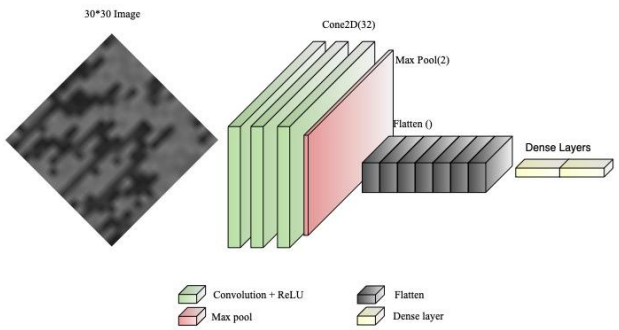

Fig. 5. The structure of the first model.

The Dropout layer (commented out) randomly drops a certain percentage (25% in this example) of activations during training. This helps prevent the model from overfitting to the training data by forcing it to learn more robust features.

After the convolutional and pooling layers, the model uses a Flatten layer to convert the 3D feature maps into a 1D vector (see Fig. 6). This allows the fully-connected layers to process the extracted features. The model then uses several fully-connected layers (Dense) to classify the image. These layers work similarly to traditional neural networks, where each neuron receives input from all neurons in the previous layer, performs weighted sums, and applies an activation function. The first three fully connected layers (4096, 2048, and 1024 neurons) are responsible for learning complex, high-level representations based on the extracted features. The relu activation allows these layers to learn non-linear relationships between the features.

The final Dense layer has only one neuron with a sigmoid activation function. This neuron outputs a value between 0 and 1, representing the probability of the image belonging to a specific class. As a summary of this model. The convolutional layers act as feature detectors, extracting progressively more complex features from the input image. The pooling layers reduce the dimensionality of the data while retaining important information. The dropout layer helps prevent overfitting. The fully-connected layers learn high-level representations and produce the final classification probability.

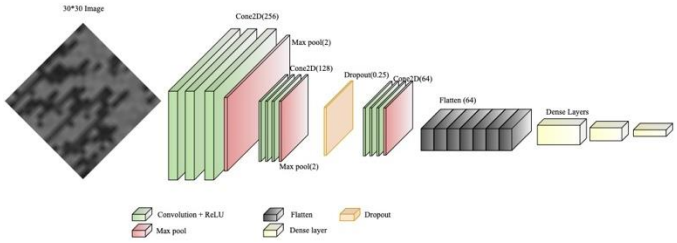

Fig. 6. The structure of the second CNN model.

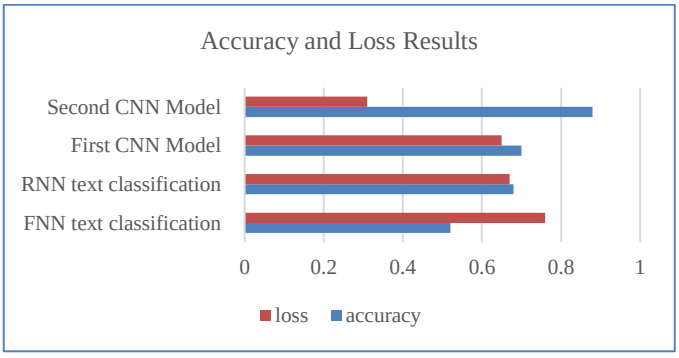

Fig. 7. Bar chart for accuracy and loss.

Using this complex model over these grayscale pictures gives accuracy rate 88% with loss 0.31. Comprehensive performance evaluation through bar charts (Fig. 7) illustrates accuracy and loss metrics for both text and image classification. The findings suggest that combining behavioral analysis with AI models, particularly in the image domain, holds promise for effective malware detection. This multimodal approach provides a holistic understanding of malware behavior, potentially enhancing overall detection capabilities in the evolving cybersecurity landscape. The study contributes to advancing malware detection methodologies by leveraging the synergy between static and dynamic analyses, bolstered by AI integration, and offers insights into the promising potential of image-based classification for improved accuracy in identifying malicious behavior.

The Second Model with numerous convolutional and fully-connected layers grants high capacity for learning intricate features. While advantageous for complex datasets, it can lead to overfitting, particularly with limited training data. The model memorizes training data too well, hindering performance on unseen examples. Furthermore, training and running this deep model can be computationally expensive due to the high number of parameters. This translates to significant processing power and memory requirements, potentially limiting its use in resource-constrained environments. The results from the text classification and image classification shows that these methods of analyzing malware might be a good way to detect the malware using the extracted behavioral features.

V. Conclusion

This study successfully employs dynamic-based analysis within a virtual machine (VM) to extract crucial behavioral features from Windows malware. Integrating these features with advanced text and image classification models (RNN and CNN) shows promise for malware detection. Image classification, based on transformed feature data, achieves a superior accuracy of 88% compared to 68% in text classification. This multi-modal approach, combining behavioral analysis with AI models, provides a nuanced understanding of malware behavior. To enhance model robustness, we recommend increasing the number of malware and benign samples, including a wider range of malware families, and exploring additional features like registry changes. Experimenting with different visualization techniques for image generation and testing more complex CNN architectures or pre-trained models with fine- tuning could further improve accuracy. Addressing adversarial attacks is crucial; incorporating noise resilience mechanisms is suggested for future work. These enhancements contribute to advancing malware detection methodologies, ensuring adaptability in the evolving cybersecurity landscape.

References

[1] Aslan, Ö., & Samet, R. (2019). A comprehensive review on malware detection approaches. IEEE Access, Advance online publication. https://doi.org/10.1109/ACCESS.2019.2963724

[2] Roundy, K.A. and Miller, B.P., 2013, August. Binary-code obfuscations in prevalent packer tools. In Proceedings of the 2013 ACM workshop on Software PROtection (pp. 3-14).M. Young, The Techincal Writers Handbook. Mill Valley, CA: University Science, 1989.

[3] Rossow, C., Dietrich, C. J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N. & van Steen, M. (2012). Prudent practices for designing malware experiments: Status quo and outlook. In 2012 IEEE Symposium on Security and Privacy (pp. 65-79). IEEE.

[4] Nafiiev, A., Kholodulkin, H., & Rodionov, A. (2022). Comparative analysis of machine learning methods for detecting malicious files. Algorithms and Methods of Cyber Attacks Prevention and Counteraction.

[5] V. S. P. Davuluru, B. N. Narayanan and E. J. Balster, "Convolutional Neural Networks as Classification Tools and Feature Extractors for

Distinguishing Malware Programs," 2019 IEEE National Aerospace and Electronics Conference (NAECON), 2019, pp. 273-277.

[6] M. Kruczkowski and E. Niewiadomska-Szynkiewicz, "Support Vector Machine for malware analysis and classification," 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014, pp. 415-420.

[7] Sharma, O., Sharma, A., & Kalia, A. (2022). Windows and IoT malware visualization and classification with deep CNN and Xception CNN using Markov images. Journal of Intelligent Information Systems. Advance online publication.

[8] Arabo, A., Dijoux, R., Poulain, T., & Chevalier, G. (2020). Detecting Ransomware Using Process Behavior Analysis. Procedia Computer Science, 168, 289-296.